

# **Analysis of Cross-Sectional Data**

## **Machine Learning Approaches to Regression**

**Kevin Sheppard**

# **Analysis of Cross-Sectional Data**

## **Best Subset Regression and Stepwise Regression**

- Best Subset Regression
- Forward Stepwise Regression
- Backward Stepwise Regression
- Hybrid Stepwise Regression

# Best Subset Regression

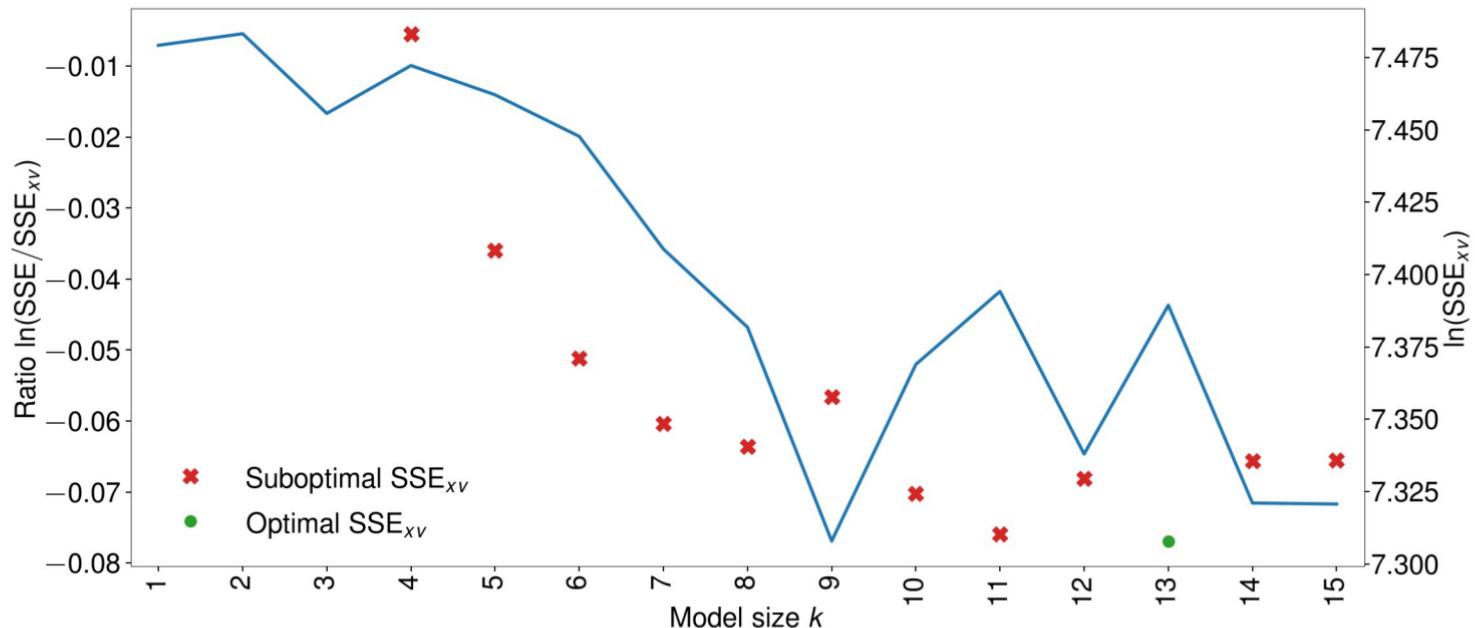
- With  $p$  candidate variables, consider all  $2^p$  possible models
- For each  $k = 1, \dots, p$  find the best model in terms of  $SSE$
- From set of  $p$  best models, select preferred model using cross-validation
- Example regression is portfolio tracking/replication

$$VWM_t = \sum_{i=1}^k \beta_i R_{i,t} + \epsilon_t$$

- $R_{i,t}$  are industry portfolio returns
- BSR example uses 15 randomly selected industry portfolios due to computational limits

# Best Subset Regression

```
In [3]: xval_plot(bsr_sse, bsr_sse_xv)
```

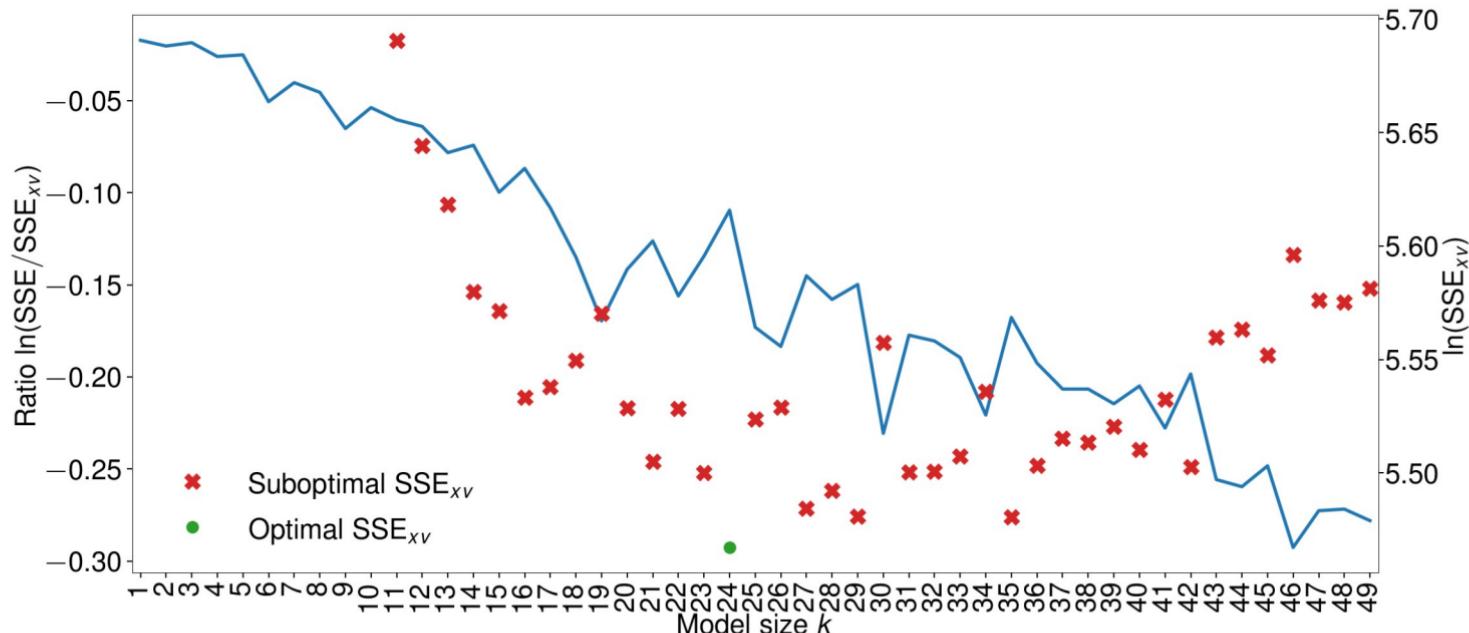


# Forward Stepwise Regression

- When  $p$  is large Best Subset Regression is computationally impossible
- Forward Stepwise Regression builds a sequence of models using two principles
  1. Start from the previous model with  $i$  variables
  2. Considering all excluded variables, **add** the variable to model  $i + 1$  that produces the **largest reduction in SSE**
- Begins with model 0 that includes no variables (or possibly a constant)
- Produces a sequence of  $p$  (or  $p + 1$  if a constant) models
- Preferred model is selected from these models using cross-validated SSE

# Forward Stepwise Regression

```
In [5]: xval_plot(fs_sse, fs_sse_xv)
```

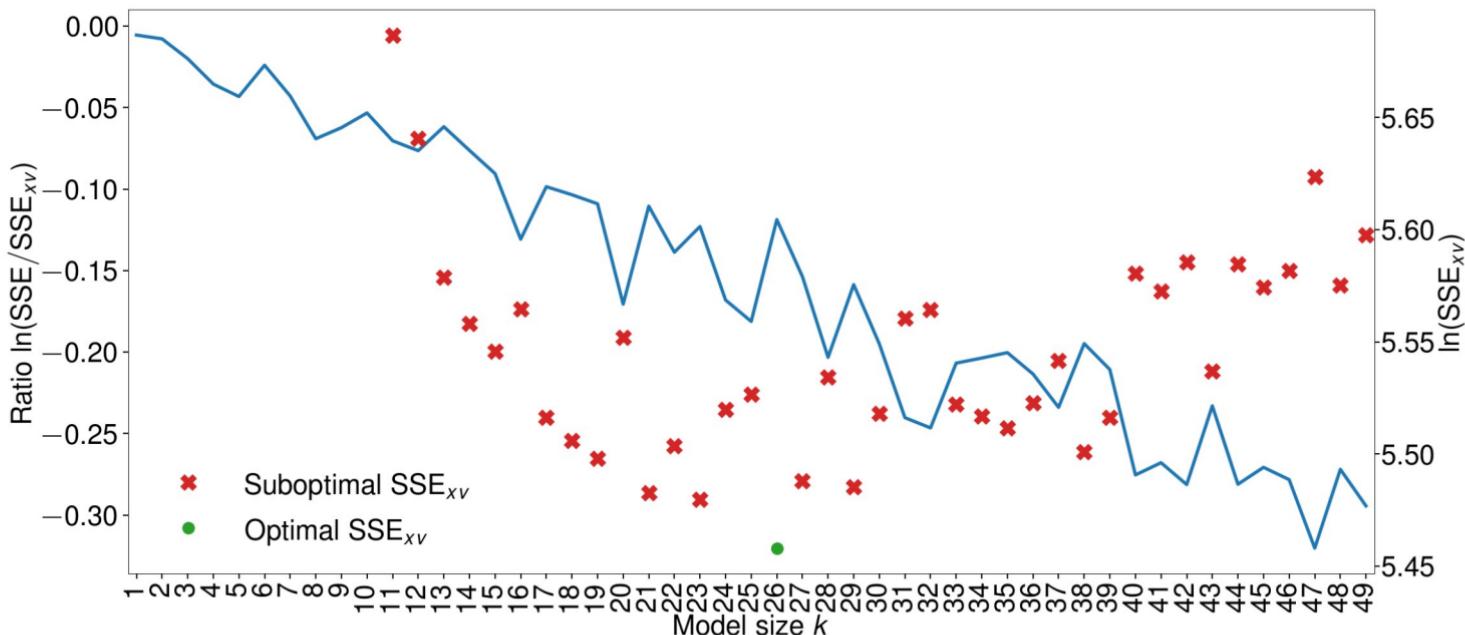


# Backward Stepwise Regression

- Similar to Forward Stepwise only constructed by *removing* variables
- Backward Stepwise Regression builds a sequence of models using two principles
  1. Start from the previous model with  $i$  variables
  2. Considering all included variables in model  $i$ , **remove** the variable that **increases the SSE the least** to construct model  $i - 1$
- Begins with model  $p$  that includes all variables
- Like FSR, produces a sequence of  $p$  (or  $p + 1$  if a constant) models
- Preferred model is selected from these models using cross-validated SSE

# Backward Stepwise Regression

```
In [7]: xval_plot(bs_sse, bs_sse_xv)
```



# Comparing Forward and Backward Stepwise Regression

```
In [8]: fs_model = fs_models[pd.Series(fs_sse_xv).idxmin()]
bs_model = bs_models[pd.Series(bs_sse_xv).idxmin()]
print(f"Number of common regressors: {len(set(bs_model).intersection(fs_model))}")
print(f"Only in FS: {' '.join(sorted(set(fs_model).difference(bs_model)))}")
print(f"Only in BS: {' '.join(sorted(set(bs_model).difference(fs_model)))}")
```

Number of common regressors: 23

Only in FS: Mach

Only in BS: Coal, Fun, Meals

# Hybrid Approaches

- Forward and Backward can be combined to produce a larger set of candidate models
- Forward-Backward
  - Use FSR to build select candidate models with  $k = 1, \dots, p$  variables
  - Starting with each of these  $p$  model, perform Backward Stepwise
- Can be iterated as much as one is willing to wait
- Might better approximate Best Subset Regression
- Final model selected from enlarged set of candidate models by minimizing the cross-validated SSE

```
In [10]: print(f"The number of models selected using forward-backward is {len(fb_model_s)}")  
print(f"Not in FS: {', '.join(sorted(set(fb_model).difference(fs_model)))}")  
print(f"Not in BS: {', '.join(sorted(set(fb_model).difference(bs_model)))}")
```

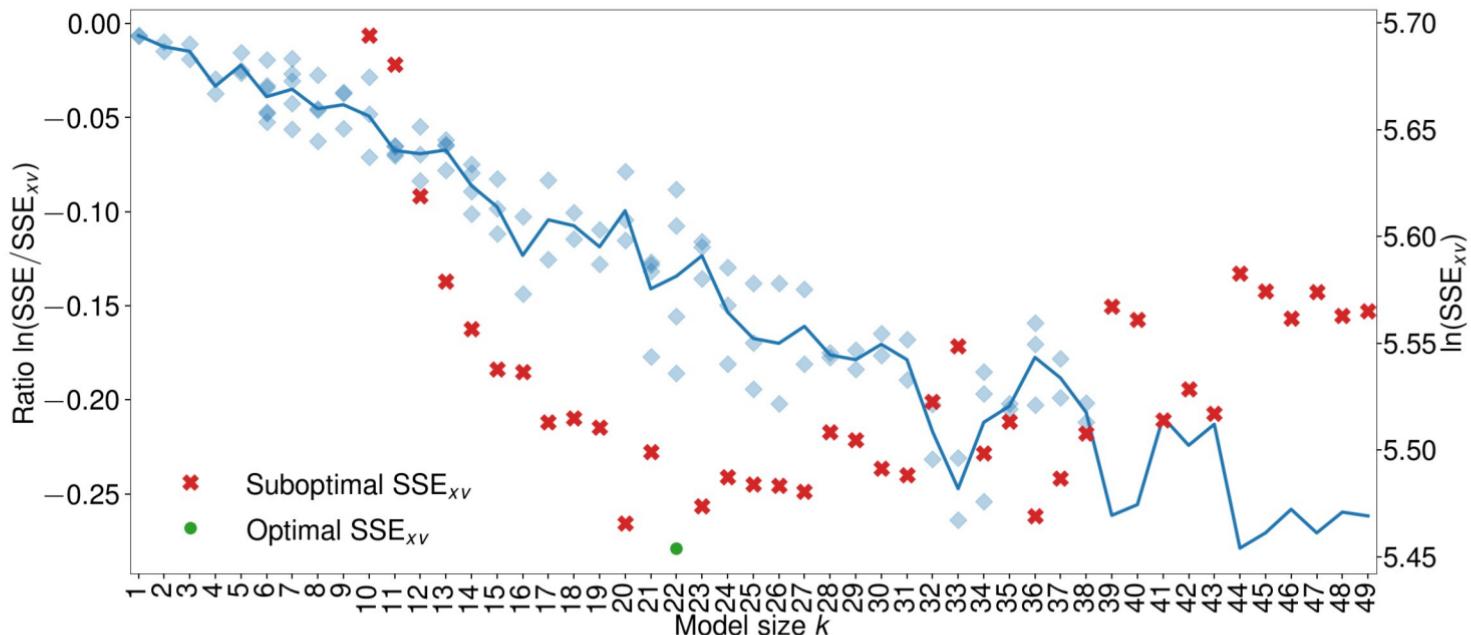
The number of models selected using forward-backward is 117

Not in FS:

Not in BS:

# Hybrid Approaches

```
In [11]: xval_plot(fb_sse, fb_sse_xv, k=fb_k)
```



# Analysis of Cross-Sectional Data

## Shrinkage Estimators

- Ridge Regression
- LASSO: Least Absolute Shrinkage and Selection Operator

**Note:** Important to standardize regressors when using shrinkage estimators

# Ridge Regression

Fit a modified least squares problem

$$\operatorname{argmin}_{\beta} (\mathbf{y} - \mathbf{X}\beta)' (\mathbf{y} - \mathbf{X}\beta) \text{ subject to } \sum_{j=1}^k \beta_j^2 \leq \omega$$

Equivalent formulation

$$\operatorname{argmin}_{\beta} (\mathbf{y} - \mathbf{X}\beta)' (\mathbf{y} - \mathbf{X}\beta) + \lambda \sum_{j=1}^k \beta_j^2$$

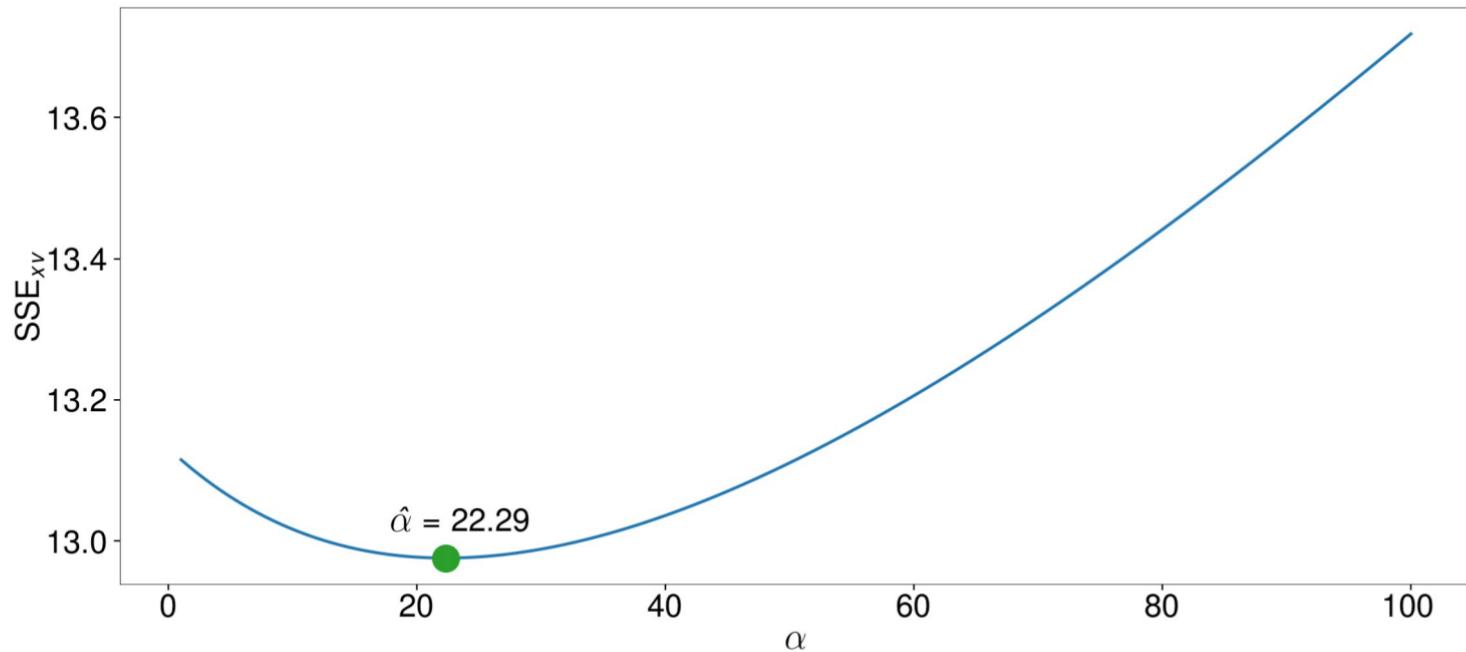
Analytical solution

$$\hat{\beta}^{\text{Ridge}} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_k)^{-1} \mathbf{X}'\mathbf{y}$$

- $\lambda$  is key shrinkage (or regularization) parameter
- Optimal  $\lambda$  is chosen using cross-validations across a grid of values

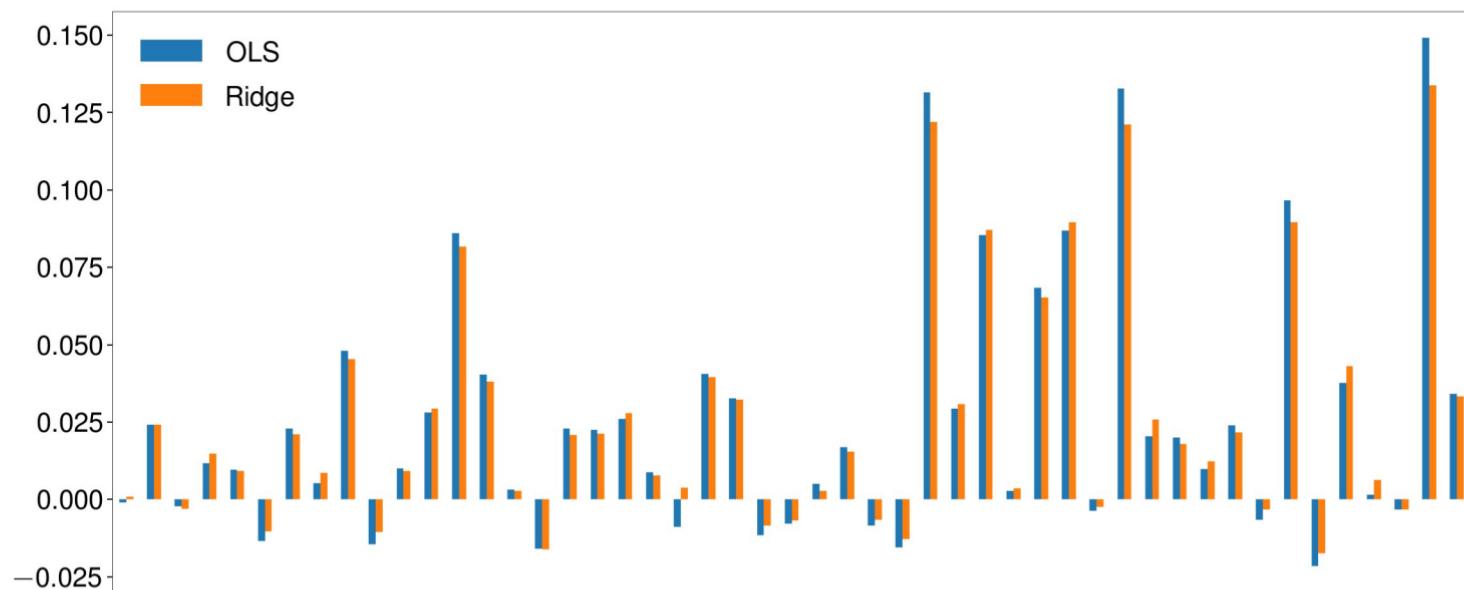
# Ridge Regression $\alpha$ Selection

```
In [13]: ridge_cv_plot()
```



# Ridge Regression

```
In [14]: ridge_coef_plot()
```



# LASSO: Least Absolute Shrinkage and Selection Operator

Similar to Ridge but with a different penalty

$$\operatorname{argmin}_{\beta} (\mathbf{y} - \mathbf{X}\beta)' (\mathbf{y} - \mathbf{X}\beta) \text{ subject to } \sum_{j=1}^k |\beta_j| \leq \omega$$

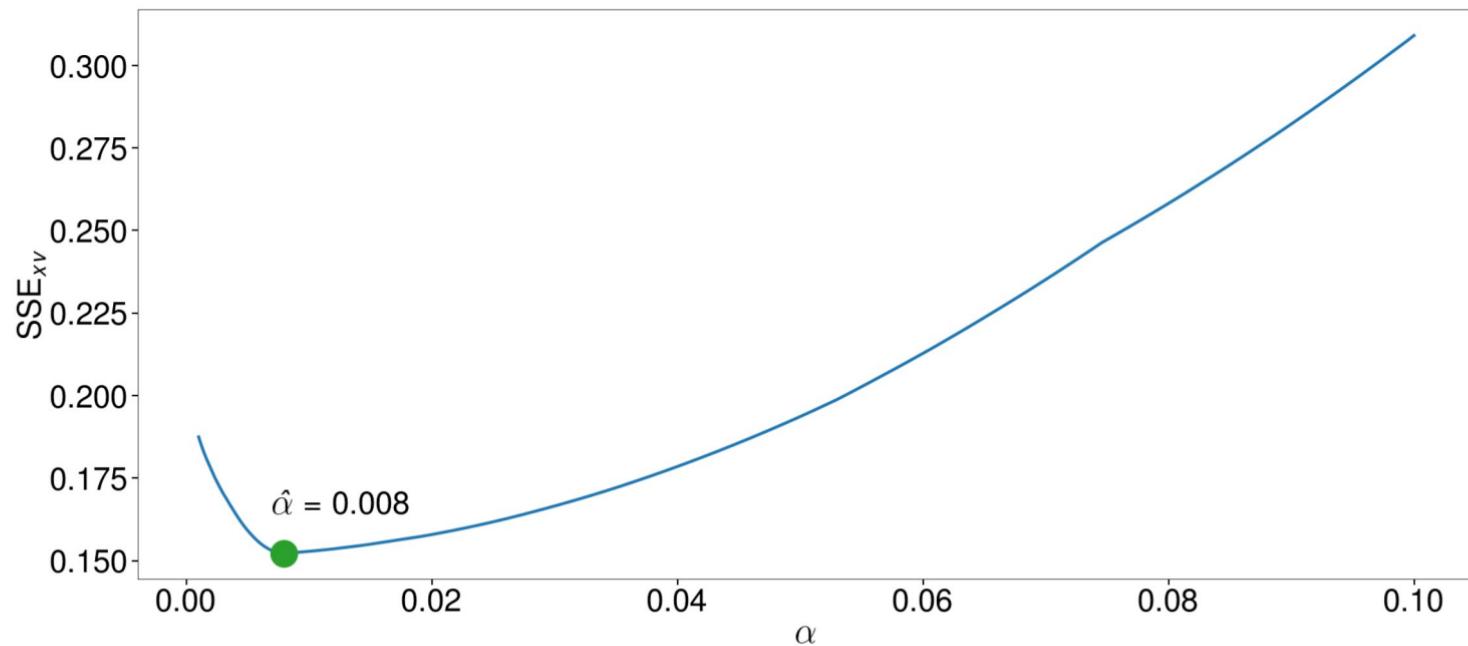
Equivalent formulation

$$\operatorname{argmin}_{\beta} (\mathbf{y} - \mathbf{X}\beta)' (\mathbf{y} - \mathbf{X}\beta) + \lambda \sum_{j=1}^k |\beta_j|$$

- Analytical solution but not easily presentable
- Has special property -- will usually estimate some coefficients to be **exactly** zero
- $\lambda$  is key shrinkage (or regularization) parameter
- Optimal  $\lambda$  is chosen using cross-validations across a grid of values

# LASSO $\alpha$ Selection

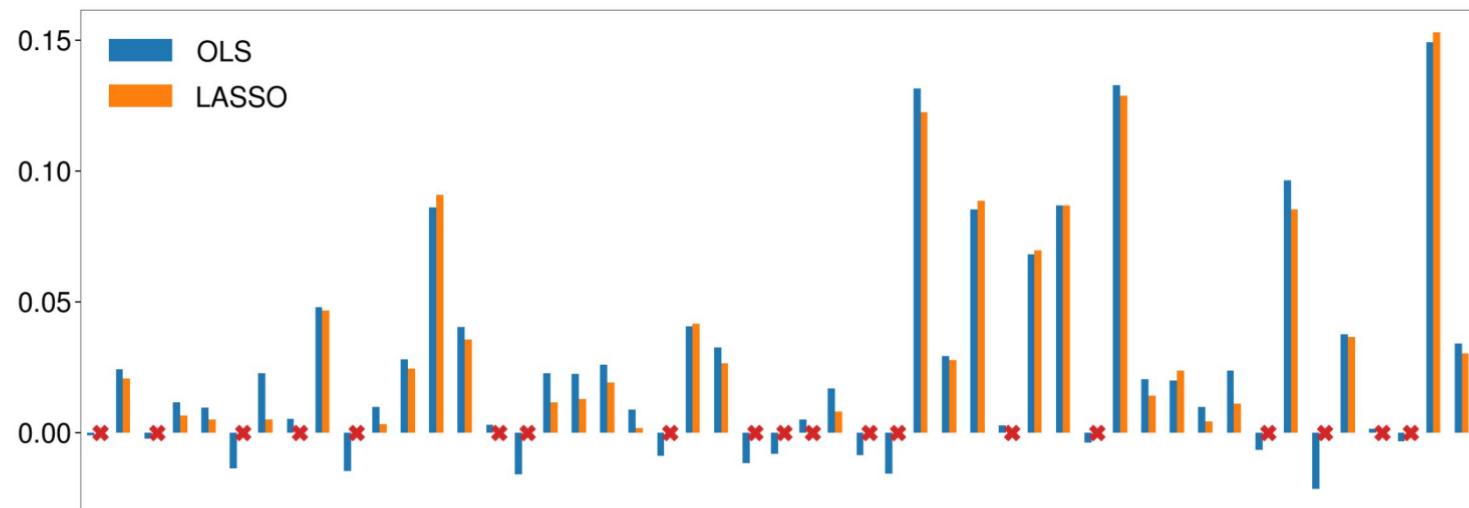
```
In [16]: lasso_cv_plot()
```



# LASSO and OLS

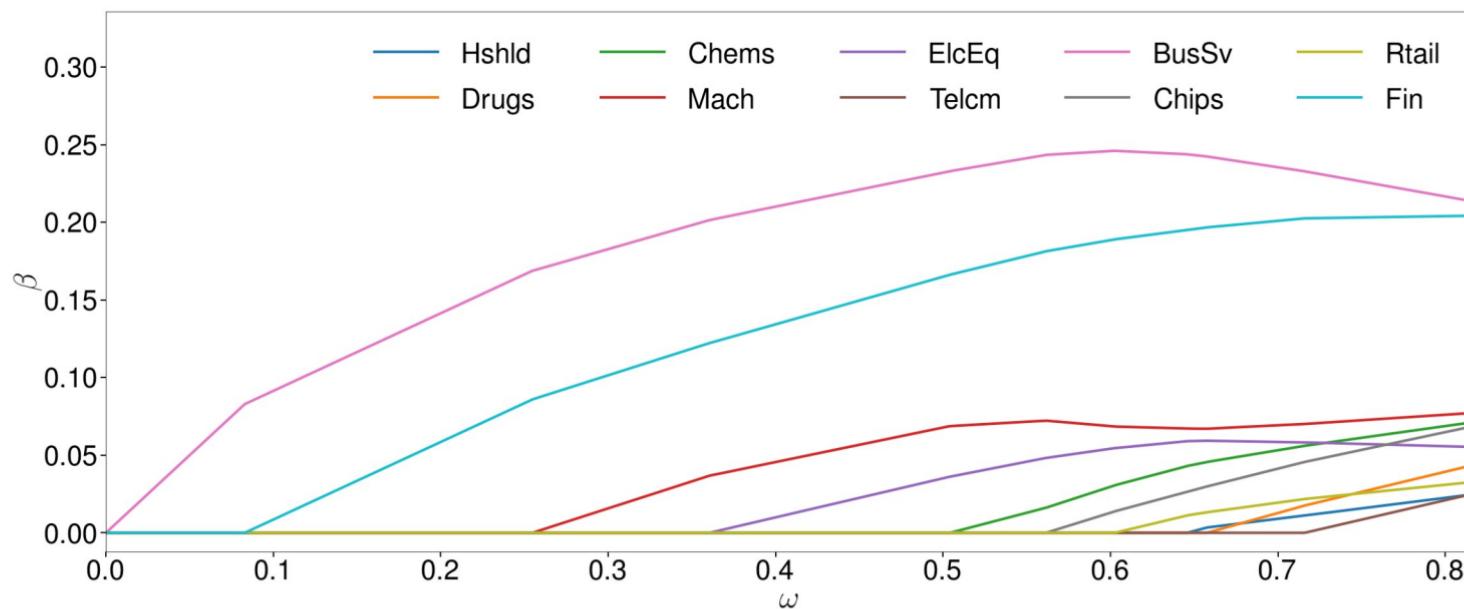
```
In [18]: lasso_coef_plot()
```

Number of non-zero coefficients: 30



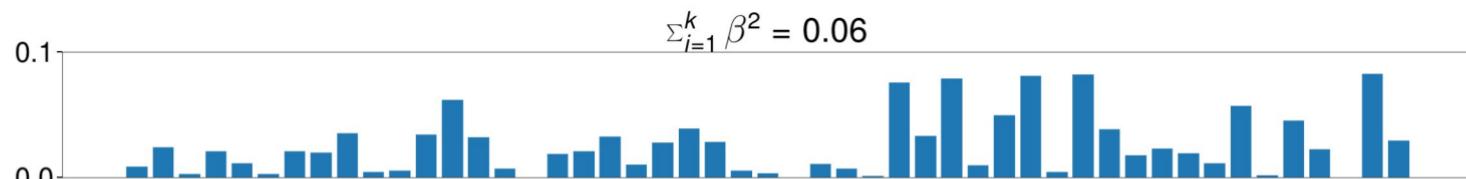
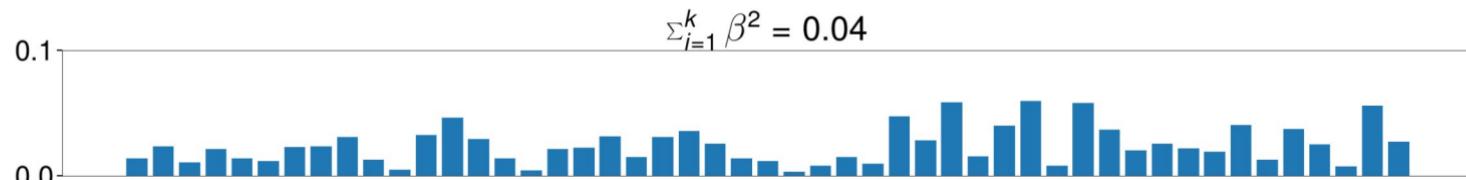
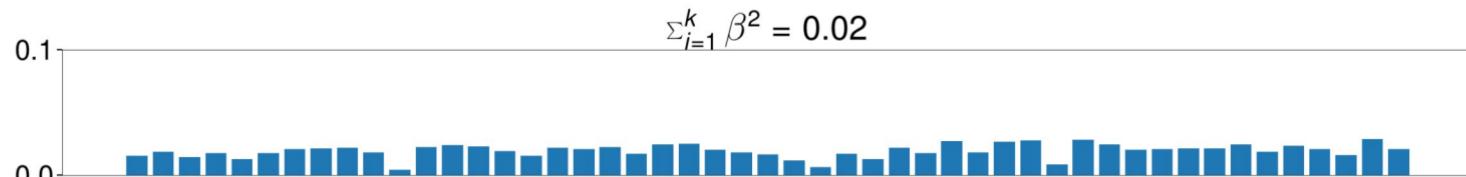
# The LASSO Coefficient Path

```
In [20]: lasso_path_plot()
```



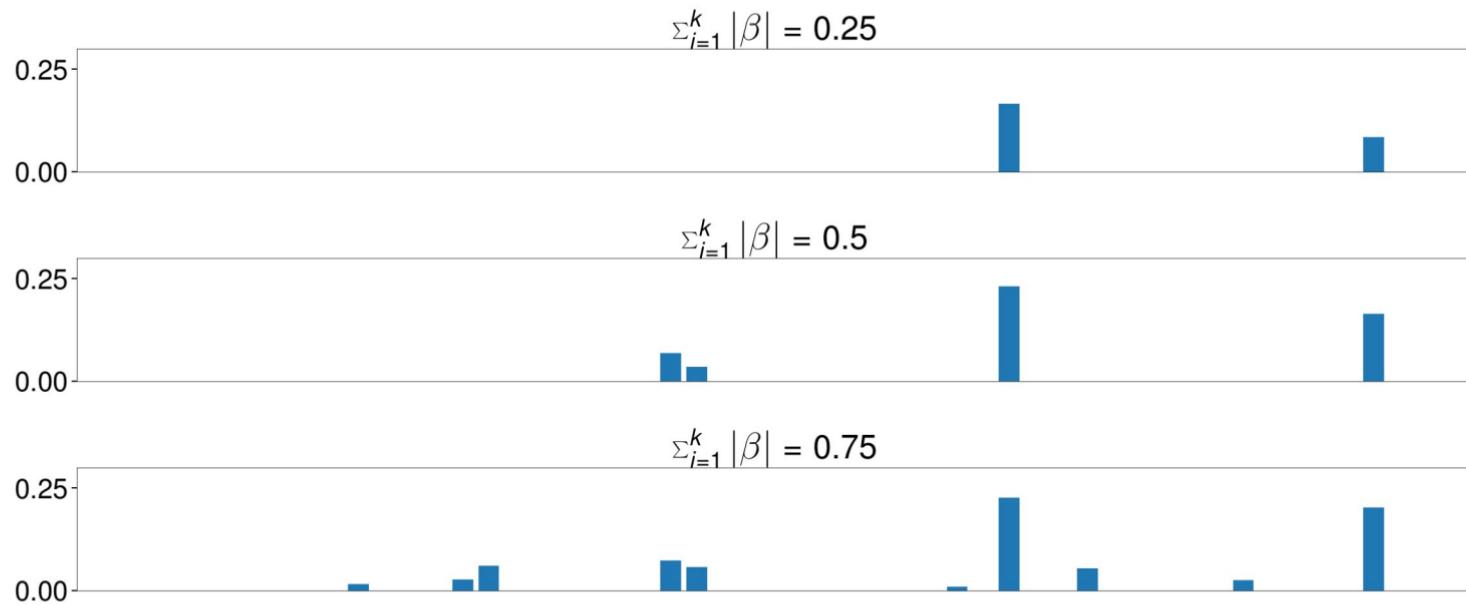
# Alternative constraints in Ridge Regression

```
In [22]: ridge_betas()
```



# Alternative constraints in LASSO

```
In [23]: lasso_betas()
```



# Analysis of Cross-Sectional Data

## Tree-based Estimators

- Regression Tree
- Model Representation
- Pruning
- Bagging and Random Forest
- Boosting

# Regression Trees

- Regression Trees build models with **indicator** variables
- Continuous regressors are transformed into Indicator variables

$$X_{i,j} \rightarrow I_{[X_{i,j} < X_j^*]}$$

- Model is then build using interaction of indicator variables
- Leads to a tree structure where the tree *splits* on each indicator variable

## Example

- 2-level tree that splits on the market first and then on the return of the size

$$I_{[VWM < 0]} I_{[SMB < 1.2]}$$

- Parameter estimates are simply averages of  $Y_i$  values where indicators are true

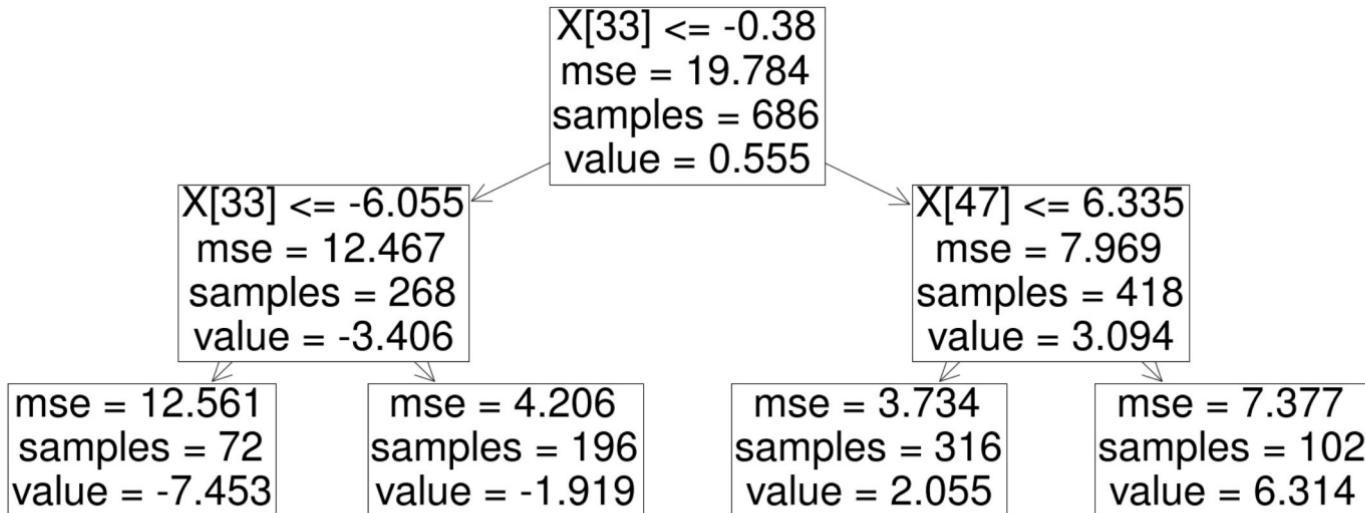
$$\frac{1}{m^*} \sum_{i=1}^n Y_i (I_{[VWM_i < 0]} I_{[SMB_i < 1.2]})$$

- $m^* = \sum_{i=1}^n I_{[VWM_i < 0]} I_{[SMB_i < 1.2]}$  counts the observation in this branch

# Regression Trees

## Two level depiction

```
In [43]: show_tree()
```



# Regression Trees

## Model Representation

- 2-level tree is regression model

$$\begin{aligned}VWM_i = & \beta_1 I_{[X_{i,33} \leq -0.38]} I_{[X_{i,33} \leq -6.055]} \\& + \beta_2 I_{[X_{i,33} \leq -0.38]} I_{[X_{i,33} > -6.055]} \\& + \beta_3 I_{[X_{i,33} > -0.38]} I_{[X_{i,47} \leq 6.335]} \\& + \beta_4 I_{[X_{i,33} > -0.38]} I_{[X_{i,47} > 6.335]} \\& + \epsilon_i\end{aligned}$$

# Regression Trees

## Key Parameters

Optimize by choosing one or more of:

- Max depth
- Maximum number of nodes
- Minimum decrease in SSE to add a new node
- Minimum group size to allow further splitting

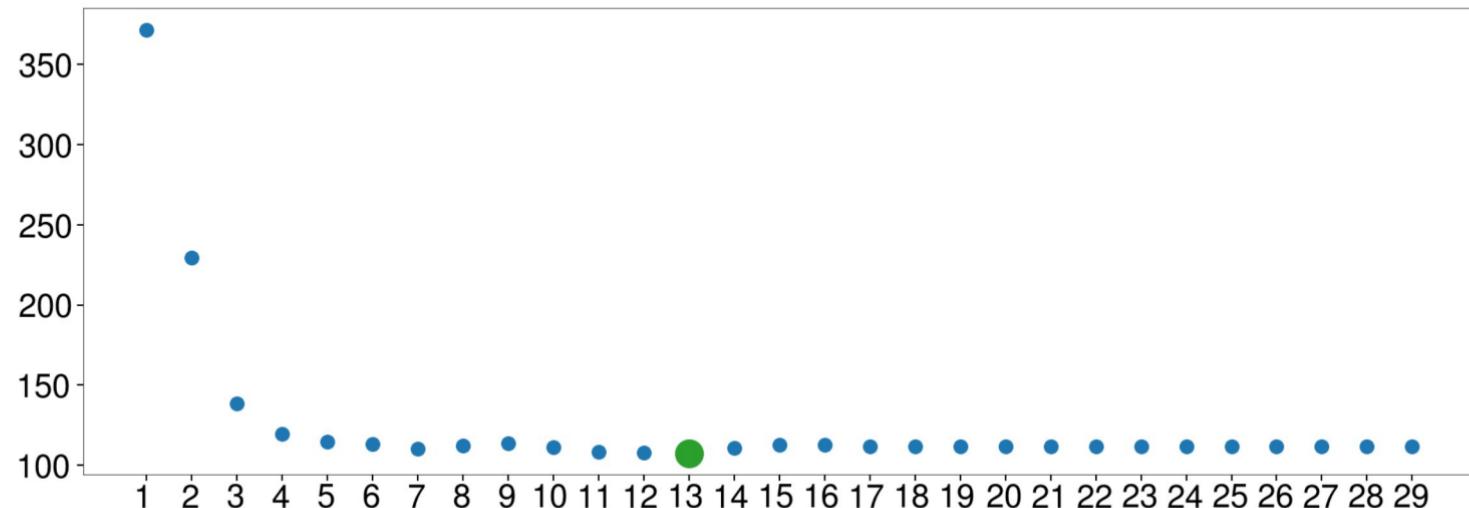
Optimize choice using cross-validation

# Regression Trees

## Max depth

```
In [46]: depth_cross_val()
```

Number of leaves: 640

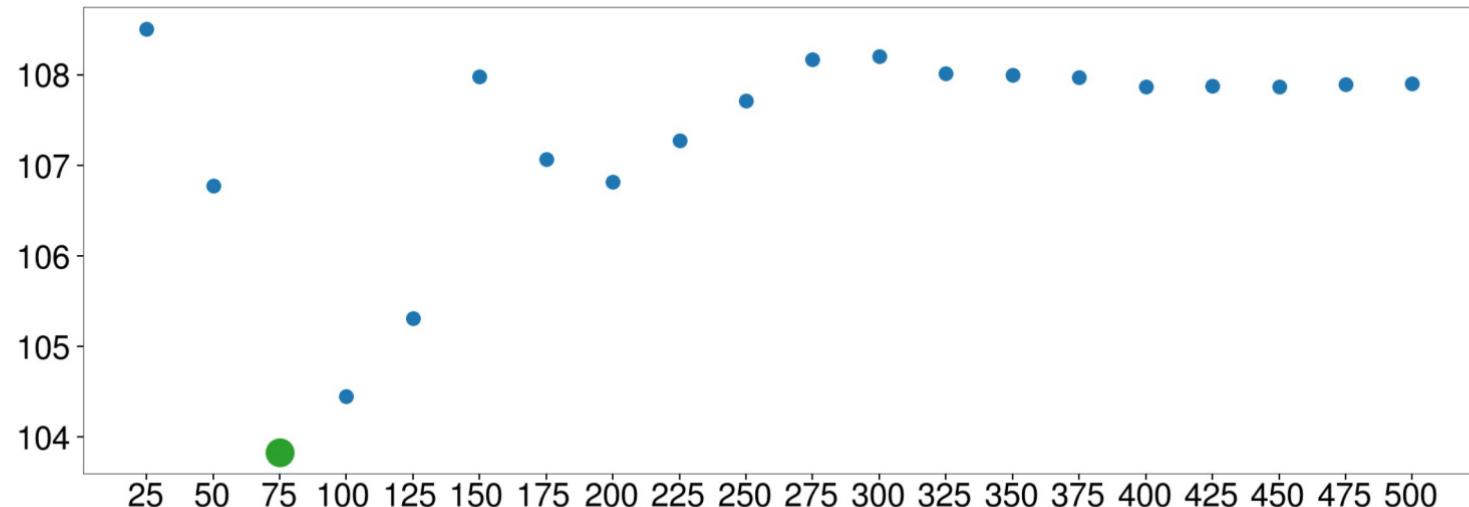


# Regression Trees

## Max nodes

```
In [47]: leaves_cross_val()
```

Maximum Depth: 8



# Pruning

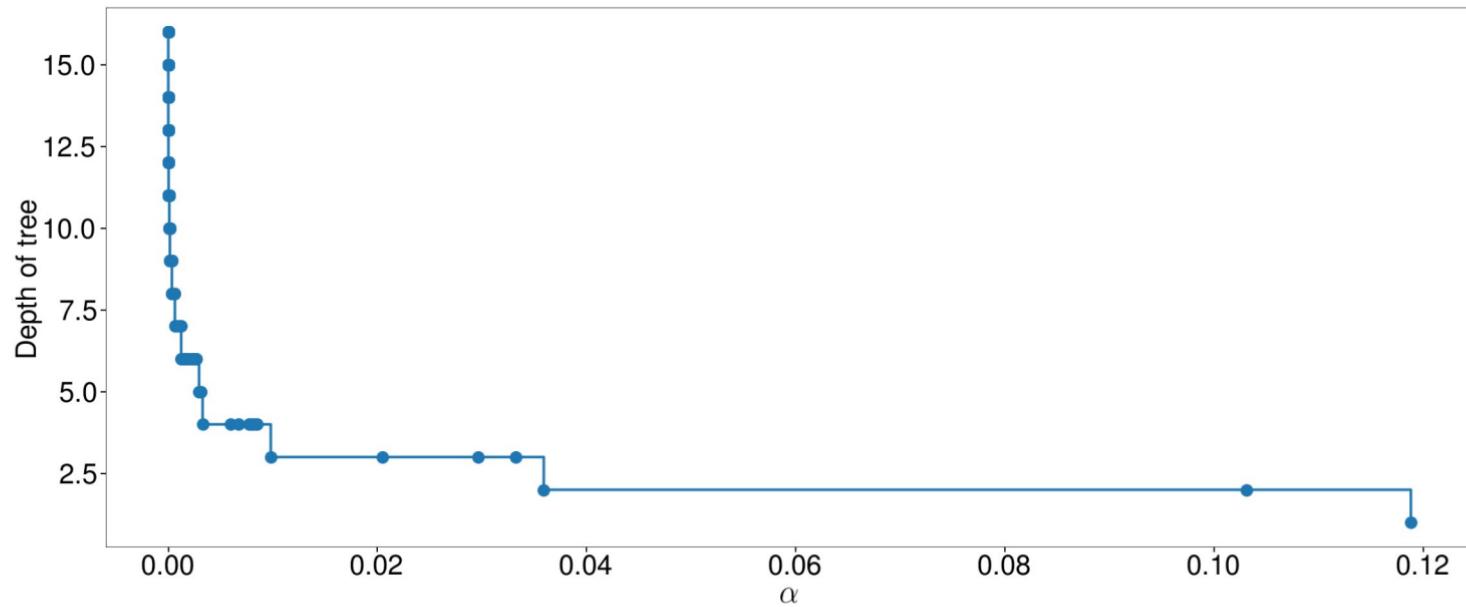
- Pruning can improve trees by removing nodes that make small improvement to the SSE
- Solve a modified objective function

$$\sum_{i=1}^n \left( Y_i - \hat{Y}_i \right)^2 + \alpha |T|$$

- $\alpha$  is a tuning parameter
- $|T|$  is the number of terminal nodes
- Larger values of  $\alpha$  reduce tree size
- Select  $\alpha$  using cross-validation by pruning a large tree for different values of  $\alpha$

# $\alpha$ and tree depth

```
In [30]: plot_tree_depth()
```

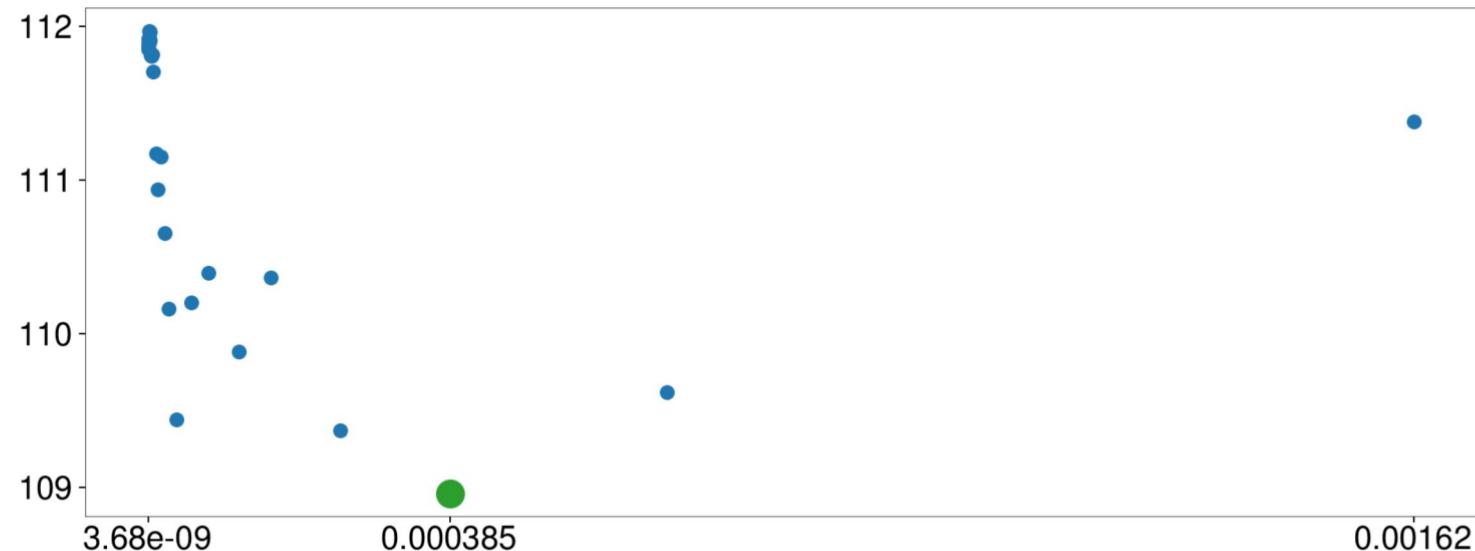


# Selecting $\alpha$ using cross-validation

```
In [32]: pruning_cross_val()
```

Number of leaves: 73

Maximum Depth: 8



# Random Forests and Bagging

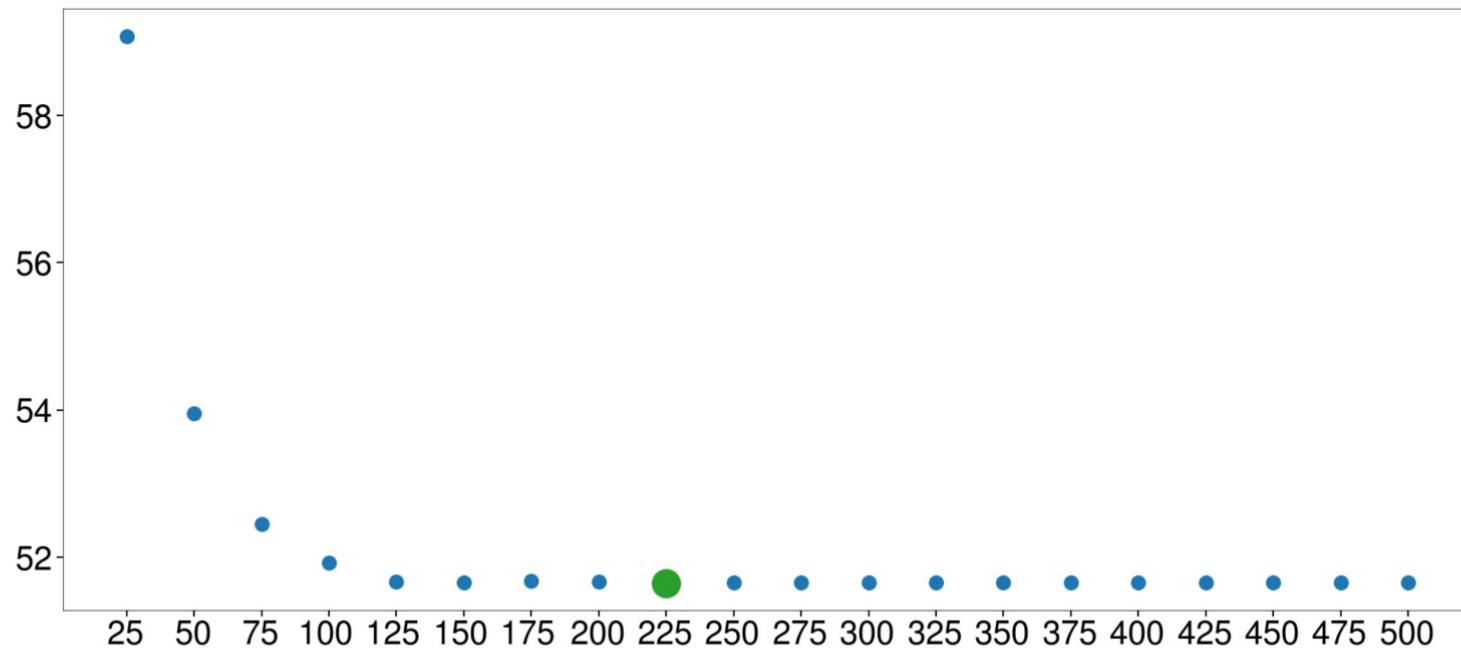
- Bagging (Bootstrap Aggregation) and Random Forests build  $B$  trees using  $B$  bootstrapped samples
- Prediction is then

$$\hat{f}^{\text{RF}}(\mathbf{x}_i) = B^{-1} \sum_{b=1}^B \hat{f}^{(b)}(\mathbf{x}_i)$$

- Random Forest improves on Bagging since:
  - Each tree is built using only  $k \approx \sqrt{p}$  of the variables
  - Produces a set of trees that are weakly correlated because most regressors are excluded from each tree
  - Used when two criteria are met
    - $p$  is large
    - A small number of strong predictors
  - **Note:** Bagging is a special case of a Random Forest when  $k = p$
- Similar tuning choices as Regression Tree plus  $B$  and  $k$

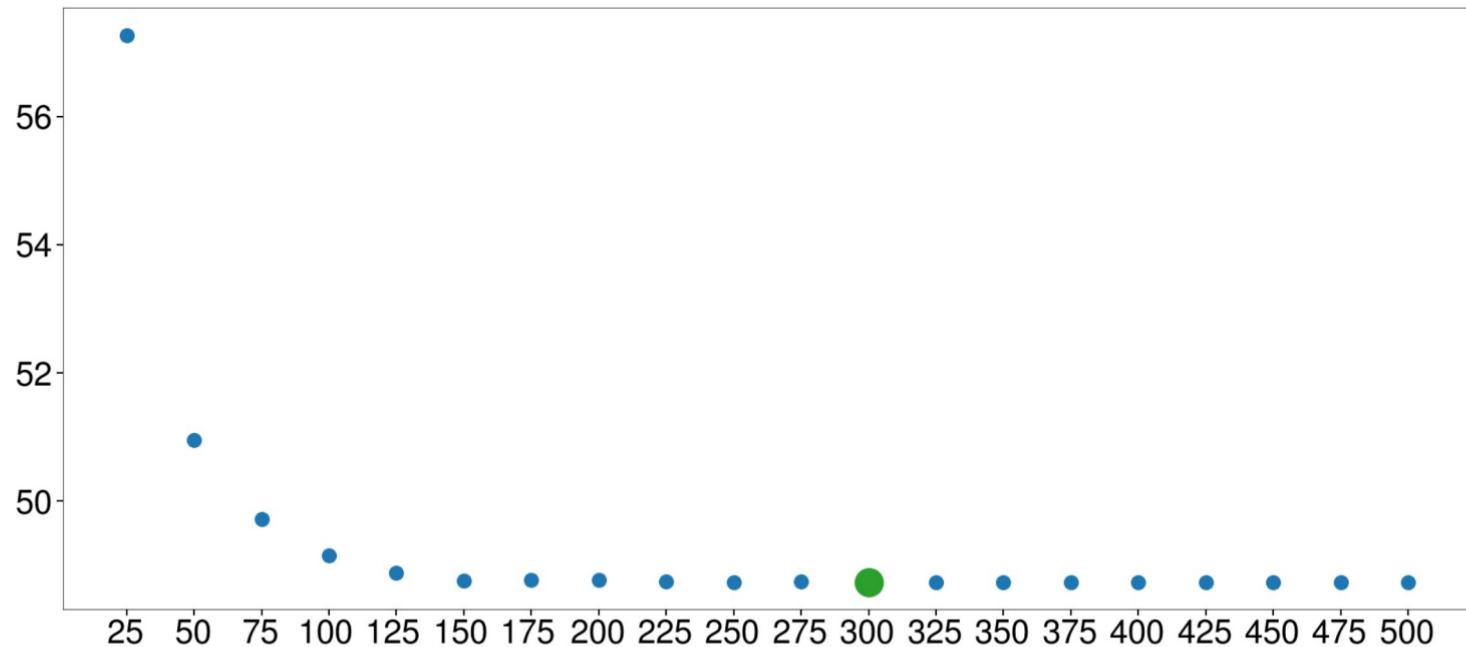
# Random Forest cross-validation

```
In [35]: rf_nodes_cross_val("sqrt")
```



# Bagging cross-validation

```
In [36]: rf_nodes_cross_val("auto")
```



# Boosting

- Boosting fits a sequence of trees each with a small number  $d$  terminal nodes
- Each tree is fit to the residuals of the previous tree
  - Child trees focus on fitting observations that were hard to fit by previous trees
  - Nodes are not added for observations that have small prediction errors
  - Building a fresh tree collects all observations in to a single leaf
- Builds models with low-interaction terms
- In a nutshell: build  $B$  (large) depth  $d$  (small) trees sequentially
- One additional parameter  $\lambda$  (small) that slows learning

$$\hat{f}(\mathbf{x}) = \sum_{b=1}^B \lambda \hat{f}^{(b)}(\mathbf{x})$$

- $\hat{f}^{(b)}$  is the prediction of the residuals produced using the previous  $b - 1$  trees

$$\hat{\epsilon}_i^{(b)} = \sum_{i=1}^{b-1} Y_i - \lambda \hat{f}^{(i)}(\mathbf{x})$$

- Three tuning parameters:  $B$ ,  $d$  and  $\lambda$

# Cross-validating Boosted Trees

In [39]:

```
boosting_cv()
```

